|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 9**

|  |  |
| --- | --- |
| **Тема Обработчики прерываний**  **Студент** Сушина А.Д.  **Группа** ИУ7-61б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Рязанова Н.Ю, |  |

Москва.

2020 г

Задание на лабораторную работу

Задание 1:

* Написать загружаемый модуль ядра, в котором зарегистрировать обработчик аппаратного прерывания с флагом IRQF\_SHARED.
* Инициализировать тасклет.
* В обработчике прерывания запланировать тасклет на выполнение.
* Вывести информацию о тасклете используя, или printk(), или seq\_file interface - <linux/seq\_file.h> (Jonathan Corber: <http://lwn.net//Articales//driver-porting/>).

Задание 2:

* Написать загружаемый модуль ядра, в котором зарегистрировать обработчик аппаратного прерывания с флагом IRQF\_SHARED.
* Инициализировать очередь работ.
* В обработчике прерывания запланировать очередь работ на выполнение.
* Вывести информацию об очереди работ используя, или printk(), или seq\_file interface - <linux/seq\_file.h> (Jonathan Corber: <http://lwn.net//Articales//driver-porting/>).

Задание 1.

Код программы представлен на листинге 1.

|  |
| --- |
| Листинг1. tasklet.c  #include <linux/module.h>  #include <linux/kernel.h>  #include <linux/init.h>  #include <linux/interrupt.h>  #include <linux/timex.h>  MODULE\_LICENSE("GPL");  static int my\_irq = 1, my\_dev\_id, irq\_cnt = 0;  void **tasklet\_function**(unsigned long data);  char my\_tasklet\_data[] = "my\_tasklet\_function was called";  DECLARE\_TASKLET(my\_tasklet, tasklet\_function, (unsigned long)&my\_tasklet\_data);  void **tasklet\_function**(unsigned long data) {  printk(KERN\_INFO "++ tasklet\_function was called. state '%ld' count '%i' data '%s'\n", my\_tasklet.state, my\_tasklet.count, (char\*)data);  }  static irqreturn\_t **my\_irq\_handler**(int irq, void \*dev) {  if (irq == my\_irq) {  printk(KERN\_INFO "++ my\_irq\_handler was called %d time(s)\n", ++irq\_cnt);  tasklet\_schedule(&my\_tasklet);  return IRQ\_HANDLED;  }  return IRQ\_NONE;  }  static int \_\_init **my\_tasklet\_init**(void) {  if (request\_irq(my\_irq, my\_irq\_handler, IRQF\_SHARED, "my\_tasklet", &my\_dev\_id)) {  printk(KERN\_ERR "++ can't get assigned IRQ %i\n", my\_irq);  return 1;  }  printk(KERN\_INFO "++ Successfully loaded handler for IRQ %d\n", my\_irq);  return 0;  }  static void \_\_exit **my\_tasklet\_exit**(void) {  tasklet\_kill(&my\_tasklet);  synchronize\_irq(my\_irq);  free\_irq(my\_irq, &my\_dev\_id);  printk(KERN\_INFO "++ tasklet unloaded, irq\_counter = %d\n", irq\_cnt);  return;  }  module\_init(my\_tasklet\_init);  module\_exit(my\_tasklet\_exit); |

Был написан обработчик прерываний для irq 1 (клавиатура).

Загрузим модуль и проверим журнал. Результаты представлены на рисунке 1.

|  |
| --- |
| Рис 1. Состояние системного журнала после загрузки модуля |

При каждом прерывании от клавиатуры вызывается обработчик прерывания my\_irq\_handler. В нем вызывается tasklet\_schedule(). При обработке тасклета выводится информация о нем.

Проверим, что my\_tasklet добавилось в список прерываний. На рисунке 2 представлено содержимое файла /proc/interrupts, что говорит о том, что my\_tasklet добавилось в список прерываний.

|  |
| --- |
| Рис 2. Содержимое файла /proc/interrupts |

Выгрузим модуль и проверим содержимое системного журнала. Результаты представлены на рисунке 3.

|  |
| --- |
| Рис 3. Состояние системного журнала после выгрузки модуля. |

Покажем, что модуль удален. На рисунке 4 представлено содержание файла /proc/interrupts, что говорит о том, что my\_tasklet удалилось из списка прерывания после выгрузки модуля.

|  |
| --- |
| Рис 4. Содержимое файла /proc/interrupts |

**Задание 2.**

Код программы представлен на листинге 2.

|  |
| --- |
| Листинг 2. queue.c  #include <linux/module.h>  #include <linux/kernel.h>  #include <linux/init.h>  #include <linux/interrupt.h>  #include <linux/timex.h>  #include <linux/workqueue.h>  MODULE\_LICENSE("GPL");  static int my\_irq = 1, my\_dev\_id, irq\_cnt = 0;  struct workqueue\_struct \*wq;  static void **my\_wq\_function**(struct work\_struct \*work);  DECLARE\_WORK(my\_work, my\_wq\_function);  static void **my\_wq\_function**(struct work\_struct \*work) {  printk(KERN\_INFO "[WORKQUEUE] handler called: data '%d'", work->data);  }  static irqreturn\_t **my\_irq\_handler**(int irq, void \*dev\_id) {  irq\_cnt++;  printk(KERN\_INFO "[WORKQUEUE] my\_irq\_handler was called %d time(s)", irq\_cnt);  queue\_work(wq, &my\_work);  return IRQ\_NONE;  }  static int \_\_init **my\_workqueue\_init**(void) {  if (request\_irq(my\_irq, my\_irq\_handler, IRQF\_SHARED, "my\_int\_workqueue", &my\_dev\_id)) {  printk(KERN\_ERR "[WORKQUEUE] can't get assigned IRQ %i\n", my\_irq);  return 1;  }  printk(KERN\_INFO "[WORKQUEUE] assigned IRQ %i\n", my\_irq);  if ((wq = create\_workqueue("my\_workqueue"))) {  printk(KERN\_INFO "[WORKQUEUE] workqueue created\n");  } else {  free\_irq(my\_irq, &my\_dev\_id);  printk(KERN\_INFO "[WORKQUEUE] workqueue allocation failed\n");  return -ENOMEM;  }  printk(KERN\_INFO "[WORKQUEUE] module is now loaded\n");  return 0;  }  static void \_\_exit **my\_workqueue\_exit**(void) {  flush\_workqueue(wq);  destroy\_workqueue(wq);  printk(KERN\_INFO "[WORKQUEUE] workqueue destroyed\n");  synchronize\_irq(my\_irq);  free\_irq(my\_irq, &my\_dev\_id);  printk(KERN\_INFO "[WORKQUEUE] IRQ handler removed\n");  printk(KERN\_INFO "[WORKQUEUE] module destroyed\n");  }  module\_init(my\_workqueue\_init);  module\_exit(my\_workqueue\_exit); |

Был написан обработчик прерываний для irq 1.

Загрузим модуль и проверим журнал. Результаты представлены на рисунке 5.

|  |
| --- |
| Рис 5. Состояние системного журнала после загрузки модуля |

Проверим, что my\_int\_worqueue добавилось в список прерываний. На рисунке 6 представлено содержимое файла /proc/interrupts, что говорит о том, что my\_int\_workueue добавилось в список прерываний.

|  |
| --- |
| Рис 6. Содержимое файла /proc/interrupts |

Выгрузим модуль и проверим содержимое системного журнала. Результаты представлены на рисунке 3.

|  |
| --- |
| Рис 7. Состояние системного журнала после выгрузки модуля. |

Покажем, что модуль удален. На рисунке 4 представлено содержание файла /proc/interrupts, что говорит о том, что my\_tasklet удалилось из списка прерывания после выгрузки модуля.

|  |
| --- |
| Рис 8. Содержимое файла /proc/interrupts |

Загрузим оба модуля и посмотрим содержимое системного журнала.

|  |
| --- |
| Рис 9. Состояние системного журнала после выгрузки модуля. |

Содержание файла /proc/interrupts:

|  |
| --- |
| Рис 10. Содержимое файла /proc/interrupts |